ECE 6332 Design Review1: Register File Design Optimization with Virtual Prototyping Tool (ViPro)

Group member:

Ningxi Liu (ECE6332) Email:nl6cg@virginia.edu

Shawn Chen (ECE6332) Email:sc7cq@virginia.edu

1. Introduction:

1.1 Register file

Register file is an array of processor registers in a central processing unit, and they usually have multiple read and write ports which can realize fast operation. Register file is a significant fraction of the power budget of processors, and they account for more than 25% of total power consumption in low power applications according to Nalluri's work (2007)[1]. The cell structure, bank constitution and architecture of register file all have effects on performance and power.

1.2 ViPro - virtual prototyping tool

ViPro is a tool designed for fast, efficient optimization and generation of memory macros and subsystems targeting diverse system level requirements across multiple process technology generations. It can estimate macro level metrics before subcomponents are completed or designed, compute the effects of a low level circuit change on the overall macro, and rapidly evaluate varying prototypes of arrays using new circuits or new cell technologies[2]. ViPro is now capable of evaluate SRAM by a varieties of benchmark circuit such as SNM, WM, delay and etc. A memory generator tool can generate a SRAM array with various design points such as high performance and low power.

1.3 Realizing register file optimization with ViPro

The delay and power consumption of register file are two important issues, and they usually can't be improved at the same time which makes the optimization of register file to be difficult. As a result, the design strategy of register file should be adjusted according to different targets and plenty of simulations are needed while trying different combination of parameters. To make design easier, we can use the ViPro to setup a bunch of benchmark circuits for quickly measuring critical features of register file, and build some bank level and architecture level behavior model of register file for analyzing without actually finishing subcomponents design.

2. References review

Publications about register files design and optimization are summarized in Table 1. These strategies can be integrated into ViPro as different parameters or benchmark circuits, for example the idea showed in Ref5 that partitioning global bit-line into several local bit-lines is feasible to be included in ViPro, and so as the differential-end and single end cell structure comparison in Ref6. Details of these references are illustrated further in the following paragraphs.

Table 1 Summary of references

Items	Strategy
Ref3	Analyzed the model of delay and energy by dividing register files into parts
Ref4	Applied power switch to less active register cells
Ref5	Partitioning global bit-line into several local bit-lines
Ref6	Compared differential-end and single end read port structure
Ref7	Utilized high threshold voltage transistor in register file
Ref8	Employed yield estimation and optimization in SRAM design

Kevin Linger proposed a prototype of utilizing ViPro in register file design automation[3]. In his work, the read and write module of register file was broken into different components, such as a flip flop, decoder, peripheral circuit and bit cell. The read and write delay and energy consumption were calculated by adding each component together. The number of physical ports is regarded as one important characterization parameter, as Figure 1 shows, because it has a quadratic impact on area and influences the bit cell capacitance. ViPro is capable of generating the energy and delay's trends with changing of the read port number. The quick analysis showed in Kevin's work is a good start with applying ViPro in register file design and optimization.

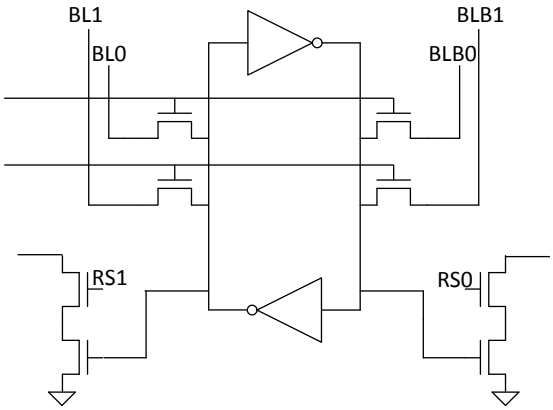

Figure 1 Register file bit cell with 2 read and 2 write ports[3]

Xuan Guan and etc proposed two techniques to reduce power consumption of register file with sacrificing delay[4]. Since the energy dissipation of bitlines is dependent on the number of registers on the bitline, which determines the average switching capacitance, the dynamic power can be reduced by

partitioning a global bitline into several local bitlines. According to previous research, 83% of power is dissipated by 25% of registers, so a drowsy technique is applied to less active registers. Figure 2 shows the structure of drowsy register cells, which can select the supply voltage with transistor C1 and C2. During run-time, the unused register files enter the drowsy state by selecting the lower supply.

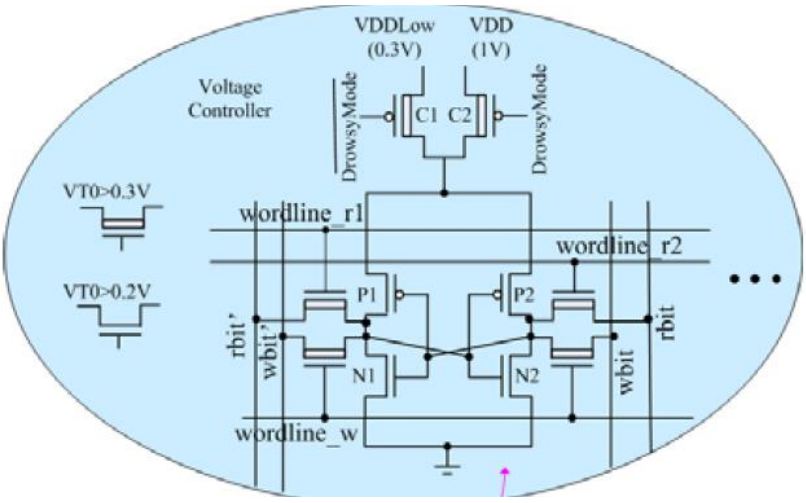

Figure 2 Basic register file structure with drowsy register cells[4]

Ataur R. Patwary and etc. compared two organization methods of local and global bit-lines of register files[5]. Register with 16 pull-downs per local bit-line and 4 pull-downs per global bit-line tends to consume less power than register with 8 pull-downs per local bit-line and 8 pull-downs per global bit-line, and they both have less normalized power consumption and read delay compared to reference with 16 pull-downs per local bit-line and 8 pull-downs. This work reveals that bit-line partition is an effective technique to reduce power and delay, and achieving one of them may result in tradeoff with the other.

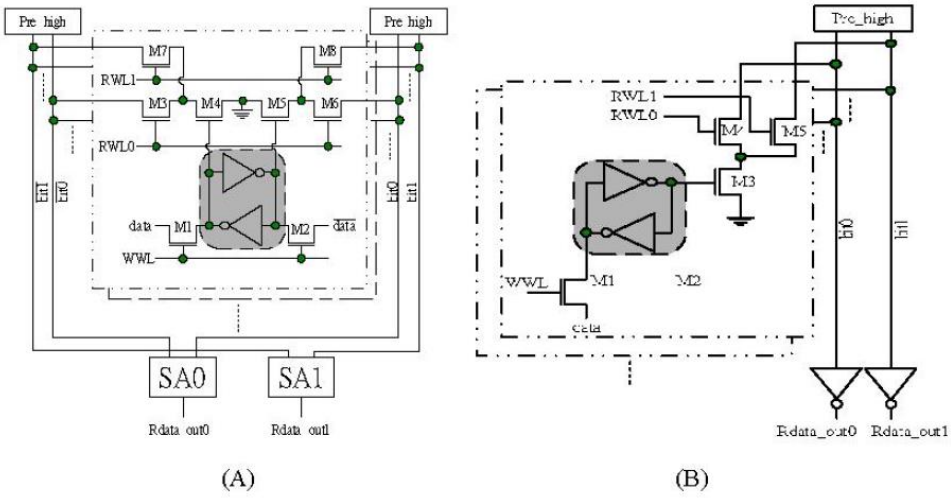

Figure 3 (A) Differential-end and (B) Single-end structure[5]

Ting-Sheng Jau and etc analyzed the difference between differential-end structure and single-end structure on power dissipation and speed[6]. According to the result, if the number of pull-downs per BL is less than 64, then it's better to use single-end structure to gain lower power consumption. However, differential-end structure must be used when pull-downs per BL excess 64, because the bit-line capacitance is too large to work in single-end structure. This work also illustrates how tradeoff between delay and power works.

Hao Yan and etc. utilized high threshold voltage transistors in register file to realize low power operation[7]. Although it only reduced 1.7% leakage power by using High Vt cell, this work introduced a method of utilizing multiple threshold voltage transistors in register file.

Jim Boley and etc. included yield constrained optimization for SRAM Design with ViPro[8]. In their work, the effect of process variation is estimated to trade-off yield with performance and energy. Parameters such as memory size, die yield and temperature are taken into account in giving the trends of dynamic read and write margin. To put yield as a standard for register file design is also a good idea, which should contains several yield models.

3. Simulations with ViPro

3.1 ViPro source code structure

The whole structure of ViPr contains a few cpp files and a user.m file and some other files. The most important file is user.m, it is a configuration for user inputs and goals. These are part of its parameters that ViPro would accept:

- Technology, used to identify the pdk
- Node, basically technology nodes without units
- Cell_Type, used to estimate bit cell height and width.
- Memory_Size
- Channel_Length
- Channel_Width
- VDD
- Word_Size
- Energy_Constraint, constraint on energy per access of the SRAM
- Delay_Constraint, constraint on the cycle time of the SRAM
- Constraint on noise margin, Ileak and bit cell area
- Bitcell size

3.2 ViPro building and calling hierarchy (ViPro - user.m - TASE - perl - ocean - spectre)

According to the source code structure, ViPro should first be compiled and get a run.exe, which is the actual executable file. Then ViPro will load user.m and parse the parameters in the setting. Once ViPro gets the input parameters, it will call TASE through system call. And TASE uses some perl scripts to call ocean and spectre to do the basic simulation tests. In the end, results would be transformed into figures and pdf reports.

3.3 Simulation results

Vipro can generate a comprehensive analysis including transistor leakage, bank mux, timing diagram of DFF, timing diagram of decoder, timing diagram of SA, yield data and etc, so only part of the simulation results will be showed.

The following simulation in ViPro is carried out under IBM 45nm technology, and configuration information of SRAM is shown below:

```
% top-level parameters
% rows and cols ignored if optimization is performed
technology = ibm45;
%wdef = 70e-9;
memSize = 1048576;
rows = 256;
banks = 32;
colMux = 4;
wordSize = 32;
vdd = 1.0;
%vsub = 1.0;
temp = 25;
% SAoffset=BL differential required for a read
SAoffset = .150;

% using ibm65 bitcell height and width - later get from cell generator
height = 0.34e-6;
width = 0.73e-6;

% calculateYield = 1; use yield model to calculate margined WL pulse width, else run for nominal case
calculateYield = 1;
targetYield = 0.95;   %if calculate yield=1, this value sets the targeted chip yield
writeTcrit = 2.03E-10;

% WL Boosting Info
WLBoost = 0;
WLoffset = 0.1; %100mV
```

```
energyConstraint = 0;
delayConstraint = 1e-9;
OPTIMIZE OBJECTIVE
           ENERGY 0
           DELAY 1.15
         KNOBS
           NBANKS 1 32
           NCOLS 1 16
           NROWS 32 512
```

Figure 4 plots the subthreshold leakage of Pass-gate in SRAM under different VDS, Figure 5 plots the timing diagram for DFF used in SRAM.

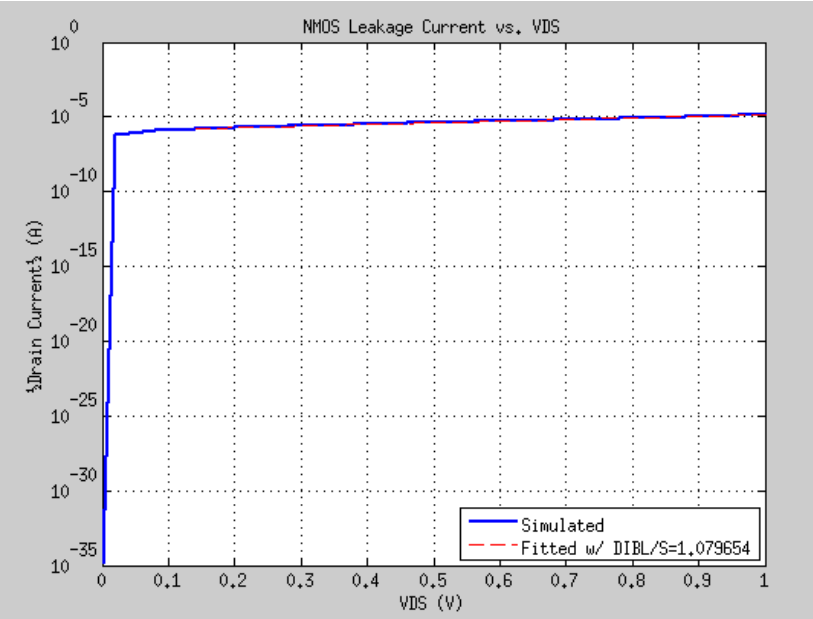

Figure 4 Leakage current under VDS of Pass-gate

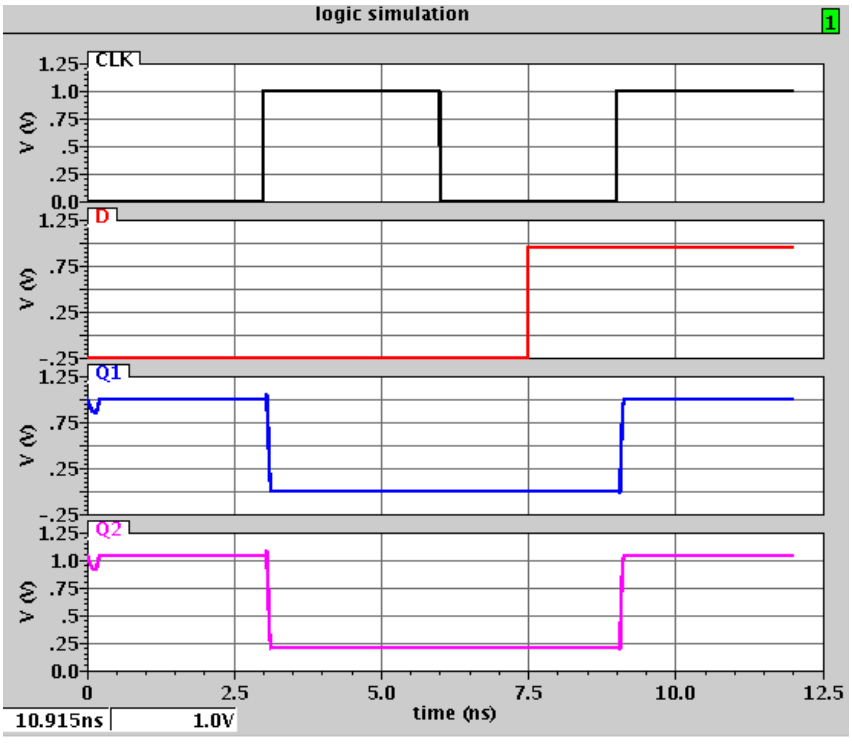

Figure 5 Timing diagram of DFF

4. Work to do

4.1 To verify the results of ViPro

ViPro is capable of evaluating the delay and energy in architecture level with the register file models, and these results are validity only if the models are correct. The best way of verifying ViPro is comparing the results with those generated from hspice. The next phase of work should be constructing a whole register file array, and to simulate it with fixed parameters such as technology, cell structure and etc. If the results of ViPro are similar with hspice, then they are accurate models; otherwise, a more precise model should substitute the existing model.

4.2 New characterization method for register file

Now there are two existing methods of characterizing a register file, which are delay and power. We need more benchmark circuits to evaluate the register file, such as yield, area and etc. Also, more parameters like process variation, technology node, transistor's size and bank organization should be taken into account. By researching latest paper, new characterization techniques and design strategies should be evaluated that if they are valuable in ViPro.

4.3 ViPro code optimization for register file

With the benefits from ViPro, fast prototyping would be easier when dealing with multiple circuits in different processes. ViPro is originally targeting the SRAM design. Since process scaling has enabled ever-larger embedded memories, scaling issues, or be more specific, variability, leakage, interconnect delay increase the difficulty in memory design. Most of the new technologies are promising but only provided in individual components. We can't ignore the effect on the whole system if we make something new. Hence, we introduce this methodology of generating and evaluating prototypes of the design process. ViPro is therefore a excellent tool allowing us to explore.

Now the source code of ViPro is accessible and bug-clear. With the source code in hand, we can first compile and try the process with SRAM. If the results are as expected, we can go further replacing the SRAM design files with our register files. The integration of register files and ViPro could cause a lot of problems, but still feasible. Once we success in running virtual prototyping for register files, then it's open and free to try to improve the design of register files.

4.4 Expectation for output

Goal to develop ViPro is making it efficient and powerful software for optimization. When we finish the project, the expectation of output is a Virtual Prototyping tool which can support quick and comprehensive analysis of register file, and helps to provide an optimized register file according to specific application.

References

[1] Rakesh Nalluri and etc. Customization of Register File Banking Architecture for Low Power. VLSID 2007.
[2] Benton Calhoun. VIRTUAL PROTOTYPING (VIPRO) TOOL FOR MEMORY SUBSYSTEM DESIGN EXPLORATION AND OPTIMIZATION.
[3] Kevin Linger. Automating the Design of Register Files in VLSI Chips: A Prototyping Tool.
[4] Xuan Guan and etc. Reducing Power Consumption of Embedded Processors through Register File Partitioning and Compiler Support. ASAP 2008, P269-274.
[5] Ataur R. Patwary. Bit-Line Organization in Register Files for Low Power AND HIGH-PERFORMANCE APPLICATIONS. ICECE 2006, P505-508.
[6] Ting-Sheng Jau and etc. Analysis and Design of High Performance, Low Power Multiple Ports Register Files. APCCAS 2006, P1453-1456.
[7] Hao Yan and etc. A Low-power 8-Read 4-Write Register File Design. APCPR 2010. Page 178-181.
[8] Jim Boley and etc. Virtual Prototyper (ViPro): An SRAM Design Tool for Yield Constrained Optimization. TVLSI 2014.